Amendments to Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A method of sharing a memory module between a plurality of processors comprising:

dividing the memory module into at least two banks, ~~into n banks, where n = at least 2,~~ wherein each bank can be accessed by one or more processors at any one time;

dividing each bank into at least one block, wherein each block can be accessed by one of the plurality of processors at any one time;

mapping the memory module to allocate sequential addresses to blocks in alternate banks of the memory; [[and]]

storing data bytes in memory, wherein ~~said~~ the data bytes in sequential addresses are stored in blocks in alternate banks due to the mapping of the memory[[.]]; and

synchronizing the processors to access blocks in different banks in response to a detected memory access conflict, which is caused by at least two of the processors accessing the same of the blocks at the same time.

2 – 4. (canceled)

5. (currently amended) The method of claim 1[[4]] further including a step of determining access priorities of the processors when memory access conflict occurs.

6. (original) The method of claim 5 wherein the step of determining access priorities comprises assigning lower access priorities to processors that have caused the memory conflict.

7. (original) The method of claim 5 wherein the step of determining access priorities comprises assigning lower access priorities to processors that performed a jump.

8. (currently amended) The method of claim 1[[4]] wherein the step of synchronizing the processors comprises locking processors with lower priorities for one or more cycles when memory access conflict occurs.

9. (currently amended) A system comprising:

a plurality of processors;

a memory module comprising n banks, where n = at least 2, wherein each bank can be accessed by one or more processors at any one time;

each bank comprising x blocks, where x = at least 1, wherein each block can be accessed by one of the plurality of processors at any one time;

a memory map for allocating sequential addresses to alternate banks of the memory module; [[and]]

data bytes stored in memory, wherein said data bytes in sequential addresses are stored in alternate banks according to the memory map[[.]]; and

a flow control unit for synchronizing the processors to access blocks in alternate banks upon a detected memory access conflict.

10 – 11.(canceled)

12. (previously presented) The system of claim 9 further comprising a priority register for storing the access priority of each processor.

13. (previously presented) The system of claim 9 wherein said data bytes comprise program instructions.

14. (previously presented) The system of claim 9 further comprising a plurality of critical memory modules for storing a plurality of data bytes for each processor for reducing memory access conflicts.

15. (currently amended) A method [[of]] for sharing a memory module between a plurality of processors comprising:

dividing the memory module into at least two banks, ~~into n banks, where n = at least 2,~~ enabling the memory module to be accessed by one or more processors simultaneously;

dividing each bank into at least one block, wherein a block can be accessed by one of the plurality of processors at any one time;

mapping the memory module to allocate sequential addresses to blocks in alternate banks of the memory;

storing data words in memory, wherein data words in sequential addresses are stored in alternate banks due to the mapping of the memory; [[and]]

providing a first signal path, the first signal path coupling a cache to a processor and the memory module when selected, the cache enabling the processor to fetch a plurality of data words from different banks simultaneously[[.]];

determining whether contention has occurred, wherein two or more processors are accessing the same address range at any one time; and

synchronizing the processors to access different banks when contention has occurred.

16 – 17.(canceled)

18. (currently amended) The method of claim 15[[17]] wherein the address range coincides with at least one block.

19. (canceled)

20. (currently amended) The method of [[the]] claim 15 further including the step of providing a second signal path, the second signal path coupling the processor to the memory module when selected.

21. (currently amended) The method of [[the]] claim 20 [[15]] further including a step of activating the second signal path when contention has not occurred.

22. (canceled)

23. (currently amended) The method of [[the]] claim 15 further including a step of determining access priorities of the processors when contention has occurred.

24. (original) The method of claim 23 wherein the step of determining access priorities comprises assigning lower access priorities to processors that have caused the contention.

25. (currently amended) The method of [[the]] claim 15[[19]] wherein the step of synchronizing the processors comprises inserting wait states for processors with lower priorities when contention occurs.

26. (currently amended) The method of [[the]] claim 15 further including a step of activating the first signal path when contention has occurred.

27 - 34. (canceled)